Generalized Linear Models

Tomas Nykodym
tomas@h2o.ai

H2O WORLD 2015

Outline

- GLM Intro
- Penalized GLM
  - Lasso, Ridge, Elastic Net
- H2O GLM Overview
  - basic features
  - regularization
- Cover Type Example
  - Dataset description
  - R example
- Common usage patterns
  - finding optimal regularization
  - handling wide datasets

Generalized Linear Models (1)

- Well known statistical/machine learning method
- Fits a linear model
  - link(y) = c1*x1 + c2*x + … + cn*xn + intercept
  - easy to fit
  - easy to understand and interpret
  - well understood statistical properties
- Regression problems
  - gaussian, poisson, gamma, tweedie
- Classification
  - binomial, multinomial
- Requires good features
  - generally not as powerful “out of the box” as other models (GBM, Deep Learning)

Generalized Linear Models (2)

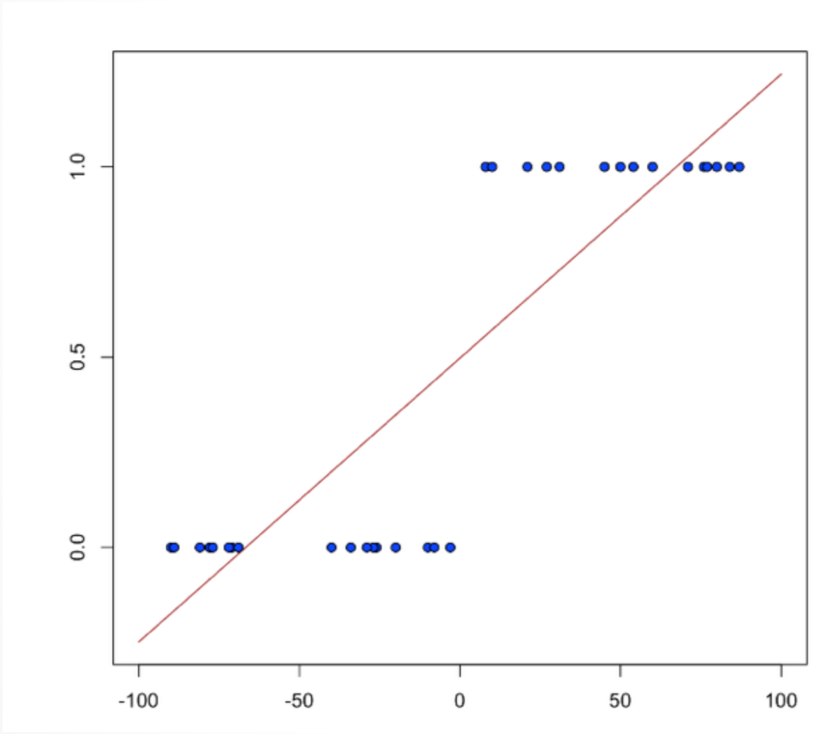
- Parametrized by Family and Link
  - Family
    - Our assumption about distribution of the response
    - e.g. poisson for regression on counts, binomial for two class classification
  - Link
    - non-linear transform of the response
    - e.g. logit to generate s-curve for logistic regression

- Fitted by maximum likelihood
  - pick the model with max probability of seeing the data
  - need an iterative solver (e.g. L-BFGS)

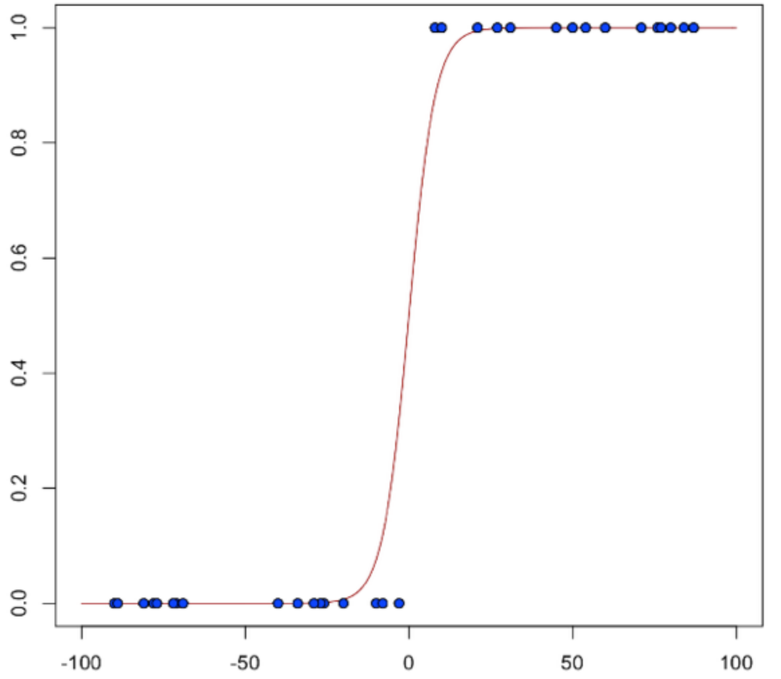
Generalized Linear Models (3)

Simple 2-class classification example

Linear Regression fit
(family=gaussian,link =identity)



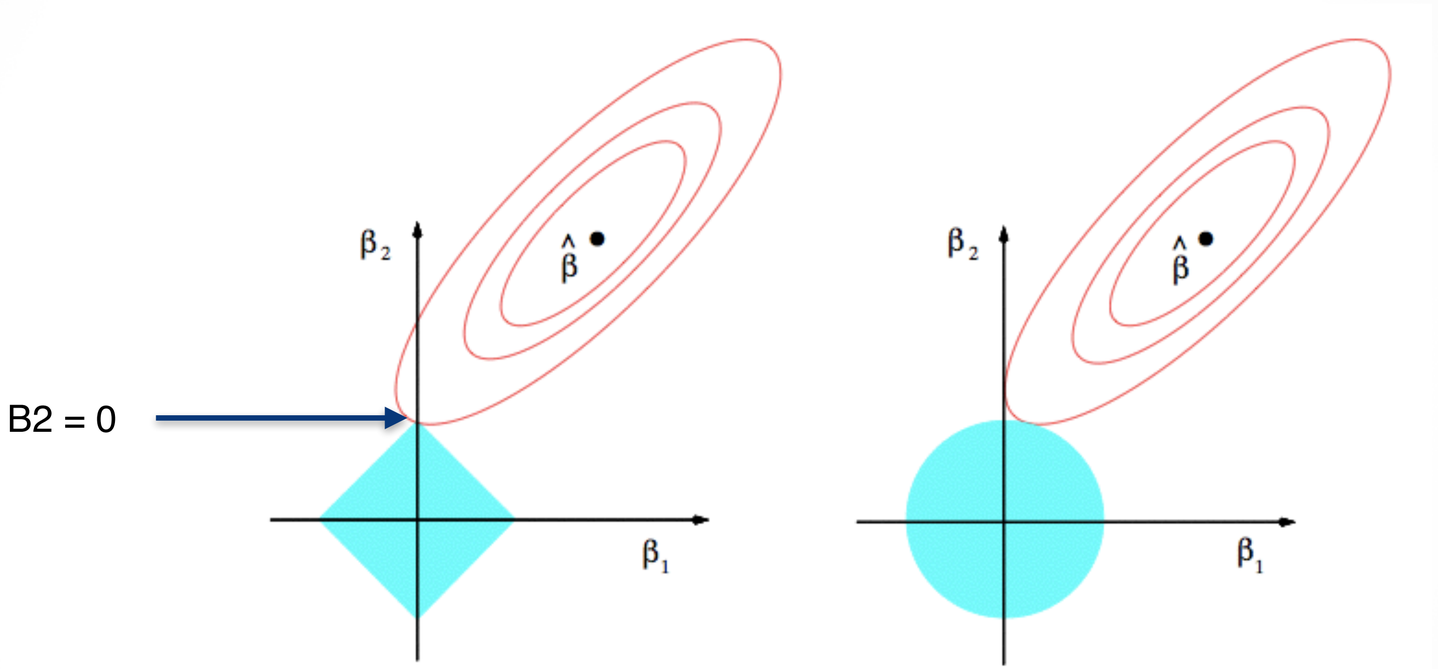
Logistic Regression fit
(family=binomial,link = logit)



Penalized Generalized Linear Models

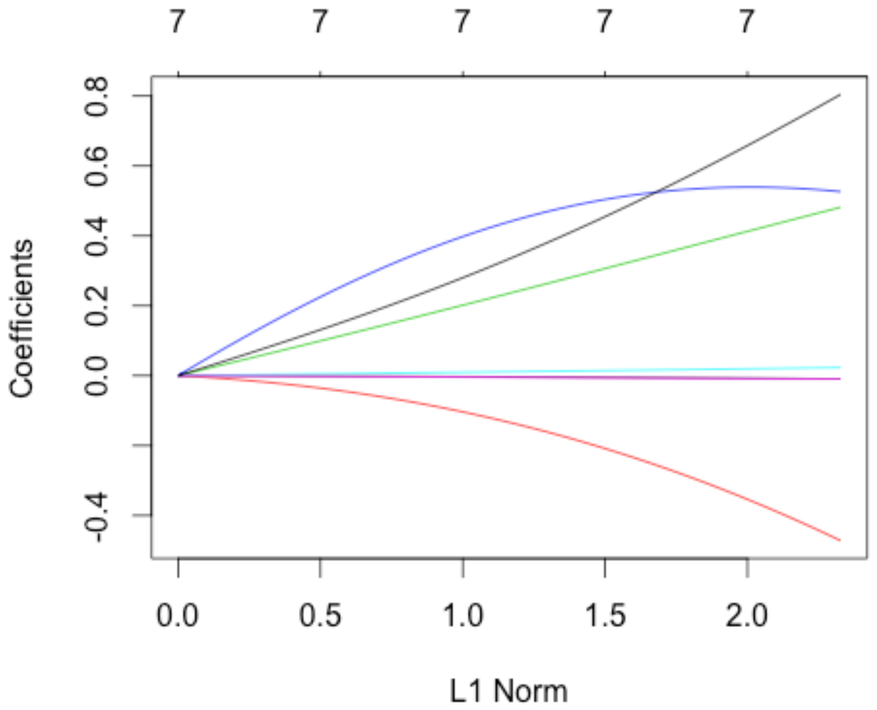
- Plain GLM is nice, but
  - can overfit (test performance way worse than training)
  - correlated variables - solution not unique, how to pick?
- Solution - Add Regularization (aka be more conservative)
  - add penalty to reduce size of the vector
  - L1 or L2 norm of the coefficient vector
- L1 versus L2
  - L1 sparse solution
    - picks one correlated variable, others discarded
  - L2 dense solution
    - correlated variables coefficients are pushed to the same value
- Elastic Net
  - combination of L1 and L2
  - sparse solution, correlated variables grouped, enter/ leave the model together

Lasso vs Ridge (1)

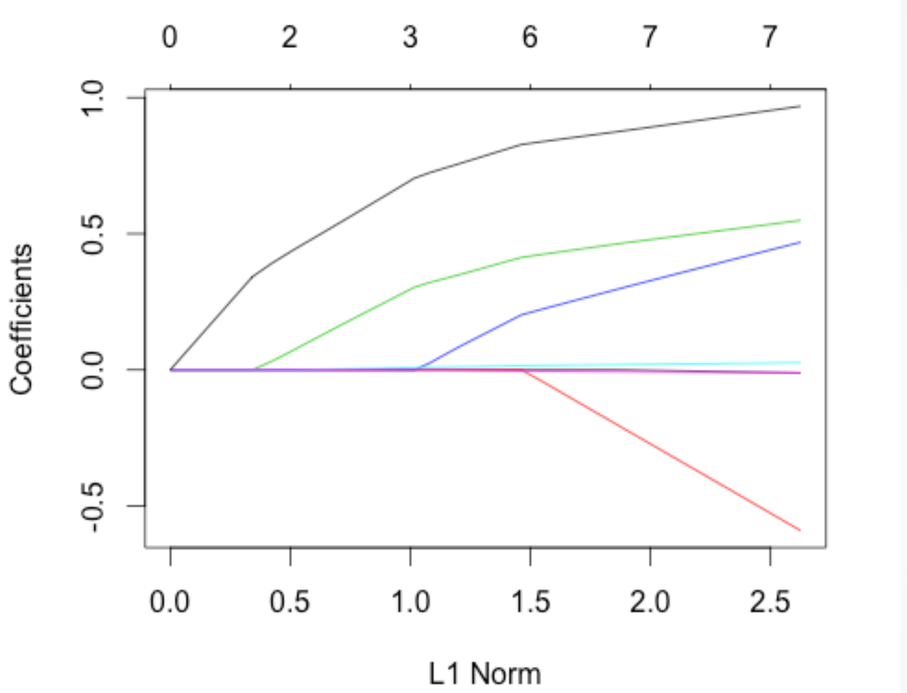


Lasso vs Ridge, source: The Elements of Statistical Learning, Hastie, Tibshirani, Friedman

Lasso vs Ridge
Regularization Path



Ridge



Lasso

H2O's GLM Overview

- Fully Distributed and Parallel
  - handles datasets with up to 100s of thousand of predictors
  - scales linearly with number of rows
  - processes datasets with 100s of millions of rows in seconds
- All standard GLM features
  - standard families
  - support for observation weights and offset
- Elastic Net Regularization
  - lambda-search - efficient computation of optimal regularization strength
  - applies strong rules to filter out in-active coefficients
- Several solvers for different problems
  - Iterative re-weighted least squares with ADMM solver
  - L-BFGS for wide problems
  - Coordinate Descent (Experimental)

H2O's GLM Overview (2)

- Automatic handling of categorical variables
  - automatically expands categoricals into 1-hot encoded binary vectors
  - Efficient handling (sparse acces, sparse covariance matrix)
  - (Unlike R) uses all levels by default if running with regularization

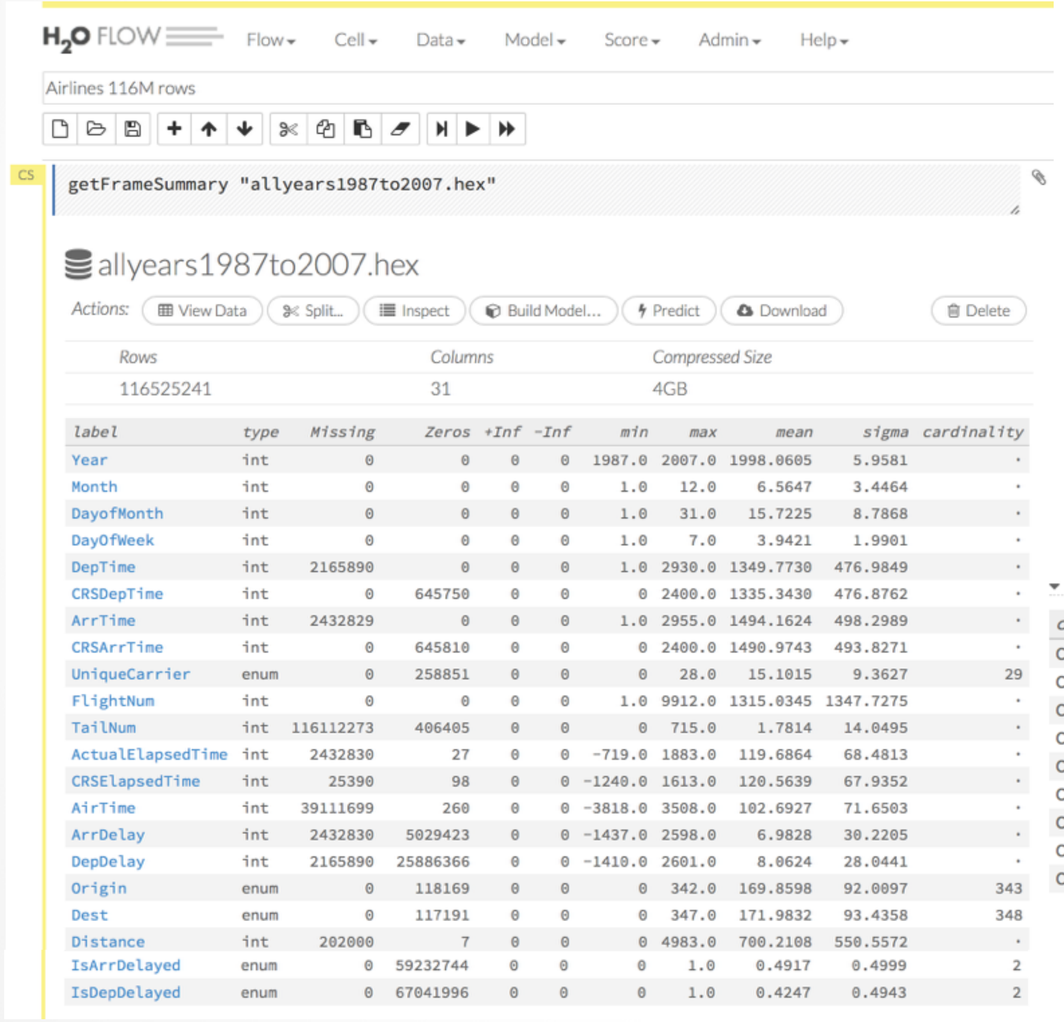
- Missing value handling
  - missing values are not handled and rows with any missing value will be omitted from the training dataset
  - need to impute missing values up front if there are many

H2O GLM Wit Elastic Net Regularization

- Standard GLM: Beta = min -loglikelihood(data)
- Penalized GLM: Beta = min -loglikelihood(data) + penalty(Beta)
- Elastic Net Penalty
  - p(Beta) = lambda * (alpha*l1norm(Beta) + (1-alpha)*l2norm2(Beta))
  - lambda is regularization strength
  - alpha controls distribution of penalty between l1 and l2
    - alpha = 0 -> Ridge regression
    - alpha = 1 -> Lasso
    - 0 < alpha < 1 -> Elastic Net
- Lambda Search:
  - Efficient search of whole regularization path
  - Build the solution from the ground up
  - Allows computation of a sparse solution of very wide datasets

Scales to “Big Data”

Airline Data: Predict Delayed Departure



allyears1987to2007.hex

Actions: View Data | Split... | Inspect | Build Model... | Predict | Download | Delete

Rows	Columns	Compressed Size
116525241	31	4GB

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality
Year	int	0	0	0	0	1987.0	2007.0	1998.0605	5.9581	·
Month	int	0	0	0	0	1.0	12.0	6.5647	3.4464	·
DayofMonth	int	0	0	0	0	1.0	31.0	15.7225	8.7868	·
DayOfWeek	int	0	0	0	0	1.0	7.0	3.9421	1.9901	·
DepTime	int	2165890	0	0	0	1.0	2930.0	1349.7730	476.9849	·
CRSDepTime	int	0	645750	0	0	0	2400.0	1335.3430	476.8762	·
ArrTime	int	2432829	0	0	0	1.0	2955.0	1494.1624	498.2989	·
CRSArrTime	int	0	645810	0	0	0	2400.0	1490.9743	493.8271	·
UniqueCarrier	enum	0	258851	0	0	0	28.0	15.1015	9.3627	29
FlightNum	int	0	0	0	0	1.0	9912.0	1315.0345	1347.7275	·
TailNum	int	116112273	406405	0	0	0	715.0	1.7814	14.0495	·
ActualElapsedTime	int	2432830	27	0	0	-719.0	1883.0	119.6864	68.4813	·
CRSElapsedTime	int	25390	98	0	0	-1240.0	1613.0	120.5639	67.9352	·
AirTime	int	39111699	260	0	0	-3818.0	3508.0	102.6927	71.6503	·
ArrDelay	int	2432830	5029423	0	0	-1437.0	2598.0	6.9828	30.2205	·
DepDelay	int	2165890	25886366	0	0	-1410.0	2601.0	8.0624	28.0441	·
Origin	enum	0	118169	0	0	0	342.0	169.8598	92.0097	343
Dest	enum	0	117191	0	0	0	347.0	171.9832	93.4358	348
Distance	int	202000	7	0	0	0	4983.0	700.2108	550.5572	·
IsArrDelayed	enum	0	59232744	0	0	0	1.0	0.4917	0.4999	2
IsDepDelayed	enum	0	67041996	0	0	0	1.0	0.4247	0.4943	2

20 years of domestic airline flight data

116M rows
31 colums
12 GB CSV
4 GB compressed

CHUNK COMPRESSION SUMMARY

chunk_type	chunk_name	count	count_percentage	size
C0L	Constant Integers	5896	6.5697	460.6 KB
C0D	Constant Reals	17930	19.9788	1.4 MB
CBS	Bits	374	0.4167	1.8 MB
CX0	Sparse Bits	5416	6.0349	3.6 MB
CXI	Sparse Integers	1532	1.7071	1.9 MB
C1	1-Byte Integers	9894	11.0246	380.6 MB
C1N	1-Byte Integers (w/o NAs)	9164	10.2112	352.3 MB
C1S	1-Byte Fractions	313	0.3488	11.9 MB
C2	2-Byte Integers	39226	43.7083	2.95 GB

Predict dep. delay Y/N

“Big Data”(2)

EC2 Demo Cluster: 8 nodes, 64 cores

H2O FLOW Flow Cell Data Model Score Admin Help

Higgs 10M rows DL

Jobs
Cluster Status
Water Meter (CPU meter)

CS getCloud

H2ODemo

CLOUD STATUS

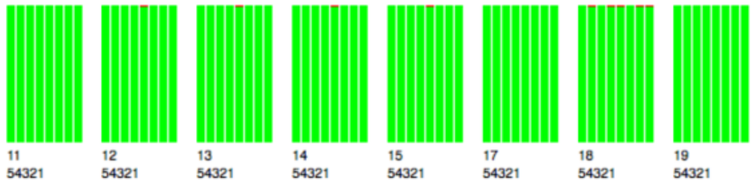
HEALTHY CONSENSUS LOCKED

Version	Started	Nodes (Used / All)
3.1.0.3013	10 hours ago	8 / 8

NODES

Name	Ping	Cores	Load	My CPU %	Sys CPU %
10.10.0.11:54321	a few seconds ago	8	7.090	100	100
10.10.0.12:54321	a few seconds ago	8	6.860	100	100
10.10.0.13:54321	a few seconds ago	8	6.820	99	100
10.10.0.14:54321	a few seconds ago	8	7.680	99	100
10.10.0.15:54321	a few seconds ago	8	6.450	99	100
10.10.0.17:54321	a few seconds ago	8	7.540	99	100
10.10.0.18:54321	a few seconds ago	8	7.680	100	100
10.10.0.19:54321	a few seconds ago	8	6.720	100	100
TOTAL	-	64	56.840	-	-

Refresh



Legend

Each bar represents one CPU.

Blue: idle time
Green: user time
Red: system time
White: other time (e.g. i/o)

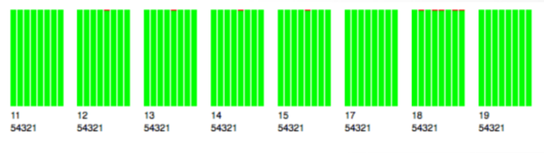
Node utilization during computation

“Big Data”

allyears1987to2007.hex

Actions: View Data | Split... | Inspect | Build Model... | Predict | Download

Rows	Columns	Compressed Size
116525241	31	4GB

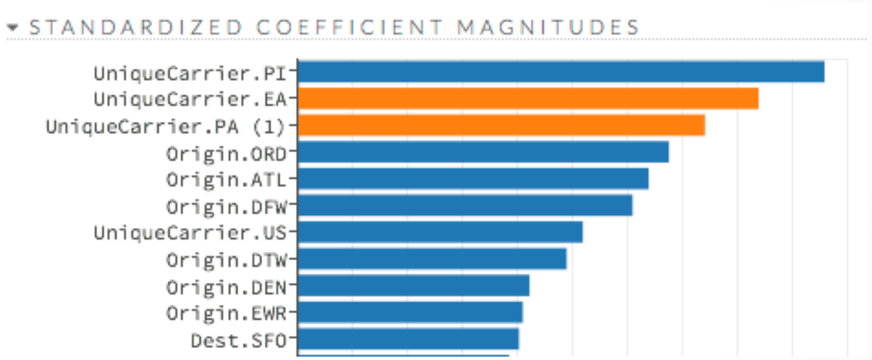


All cores maxed out

Logistic Regression: ~20s

elastic net, alpha=0.5, lambda=1.379e-4 (auto)

MSE	0.231287
r2	0.053357
logloss	0.654437
AUC	0.634441



Chicago, Atlanta, Dallas:
often delayed

Laptop Size Demo (1)

Cover Type Data:

- Predict forrest cover type based on cartographic data
- 0.5 million rows, 55 features (11 numeric + 2 categorical)
- response has 7 classes (7 cover types)
- Multinomial Problem -> 7x55 = 385 features
- Code in tutorials/glm/glm_h2oworld_demo.R(py)

Summary - Best Practices

- Regularization selection
  - if not sure, try both dense and sparse (l1, no l1)
  - don’t need to grid search alpha, only 2 or 3 values would do (e.g. 0.0, .5,.99)
  - always include some l2, i.e. set alpha to .99 instead of 1
- Wide datasets
  - datasets with ~ 10s of thousand of predictors or more
  - standard IRLSM solver does not work (can use L-BFGS)
  - IRLSM + lambda search works and is recommended!
    - (with alpha >> 0!)
    - can get solution up to low thousands of non-zeros
    - efficient way to filter out unused coefficients
  - L-BFGS + l2 penalty works
    - L-BFGS + l1 may work, but can take very long time
- Grid Search - available, not really needed
  - lambda search with 2 or 3 values of alpha is enough

Q & A